Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой

_____ О.В.Непомнящий
подпись инициалы, фамилия
« _____» _____ 2018 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 Информатика и вычислительная техника
код и наименование направления

Мобильное приложение для программы лояльности
тема

Руководитель _____ доцент, канд .тех. наук М.С.Медведев
подпись, дата должность, ученая степень инициалы, фамилия

Выпускник _____ Д.Н.Галин
подпись, дата инициалы, фамилия

Нормоконтролер _____ В.И.Иванов
подпись, дата инициалы, фамилия

Красноярск 2018

РЕФЕРАТ

Выпускная квалификационная работа по теме «Разработка мобильного приложения для программы лояльности» содержит 48 страниц, 1 список сокращений, 4 приложения. При выполнении исследовано 13 источников. Использовано 23 рисунка.

МОБИЛЬНОЕ ПРИЛОЖЕНИЕ, ПРОГРАММА ЛОЯЛЬНОСТИ, APACHE CORDOVA, IONIC FRAMEWORK, TYPESCRIPT, HTML, ANDROID, IOS.

Цель работы: написание мобильного приложения для программы лояльности

При выполнении данной работы был произведен обзор предметной области, задания на выпускную квалификационную работу, изучены существующие аналоги и сформированы требования, предъявляемые к мобильному приложению для программы лояльности.

Объект работы – мобильное приложение, поощряющее пользователей использовать определенный список услуг и магазинов.

Задачи:

- выполнить анализ существующих решений в предметной области;
- осуществить выбор программных средств моделирования и разработки мобильного приложения;
- выполнить моделирование разрабатываемого мобильного приложения;
- Выполнить программную реализацию мобильного приложения для платформ iOS, Android;
- протестировать разработанное мобильное приложение;
- проанализировать полученные результаты работы.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время рынок товаров и услуг полон однотипных по своей сути предложений от разных предпринимателей. Каждый из них заинтересован в том, чтобы покупатель, единожды воспользовавшись его услугой возвращался к нему снова и снова, а не уходил к конкуренту.

Актуальностью выбранной темы является необходимость внедрения современных технологий для повышения эффективности продаж товаров и услуг. Мобильное устройство в наши дни имеется у каждого человека, поэтому взаимодействие с ним посредством приложения, предлагающего различные бонусы и скидки является результативным методом позитивного воздействия на потребителя.

Программа лояльности в общем виде – комплекс маркетинговых мероприятий для развития повторных продаж существующим клиентам в будущем, продажи им дополнительных товаров и услуг, продвижения корпоративных идей и ценностей, других видов потенциально прибыльного поведения. Проводится, в основном, на этапе зрелости жизненного цикла товара.

Типичным примером программы лояльности компании является дисконтная карта, при дальнейших покупках с использованием дисконтной карты могут предоставляться скидки, в том числе по накопительной системе, также могут существовать системы бонусов и подарков.

Целью данной работы является разработка мобильного приложения для программы лояльности нескольких различных магазинов и услуг с широким функционалом.

1 Анализ технического задания

Необходимо разработать гибридное веб-мобильное приложение для программы лояльности, реализующее следующий функционал, различный для пользователей с разным уровнем доступа:

- авторизация;
- обзор и управление балансом
  - просмотр операций
  - перевод средств;
- личный кабинет с возможностью редактирования информации;
- доступ к магазину с возможностью потратить внутреннюю валюту или обычные рубли;
- обзор заказов, детальная информация по ним;
  - если у пользователя имеются права организации, то просмотр заказов и покупок этой организации происходит как и под обычным пользователем;
- список магазинов, участвующих в программе и детальная информация по ним;
- список акций и детальная информация по ним;
- корзина;
- ответы на часто задаваемые вопросы.
- реферальная система

Данное мобильное приложение разрабатывается для системы Elleum, модель системы – клиент-серверное взаимодействие.

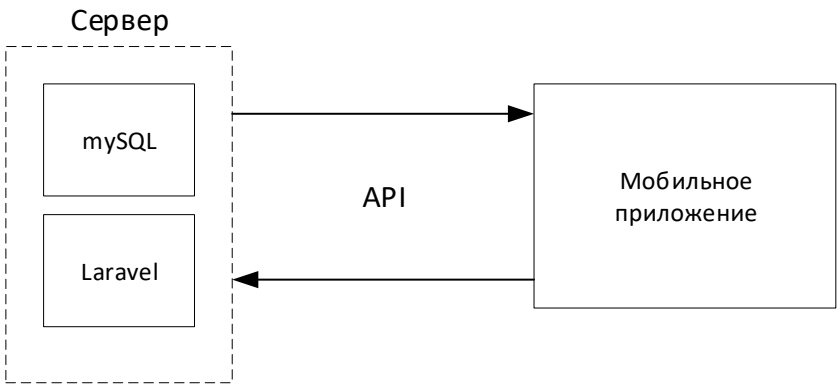

Рисунок 1 – клиент-серверное взаимодействие

Серверная часть системы работает с использованием mySQL для работы с БД и Laravel для реализации общей логики сервера. Из этого следует, что для организации работы сервера с клиентским мобильным приложением, как показано на рисунке 1, необходимо реализовать API.

1.1 Анализ технологий

1.1.1 Native приложения

Native приложения – это те программные продукты, которые разрабатывались под конкретную операционную систему. В частности, программа, написанная под ОС Android будет работать только на ней. Точно так же и с программой, написанной для ОС iOS. У каждой операционной системы есть свой набор основных процедур (API), с помощью которого она осуществляет взаимодействие с разнообразными прикладными программами. API – набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением (библиотекой, сервисом) или операционной системой для использования во внешних программных продуктах.

Нативные приложения разрабатываются с конкретным API, а, следовательно, с конкретной операционной системой.

Преимущества нативного мобильного приложения:

- позволит поддерживать высокую производительность;
- позволит обрабатывать большое количество данных на стороне клиента;
- гибкость в реализации – нативная разработка может использовать все возможности мобильной операционной системы;
- экономичность расходования аккумулятора;
- использование платежных механизмов, предоставляемые магазинами приложений, например, Play Market для ОС Android.

Одним из главных недостатков нативного мобильного приложения можно выделить трудоемкость реализации проектов что, следовательно, влияет на конечную стоимость продукта.

1.1.2 Кроссплатформенные приложения

Кроссплатформенное программное обеспечение — программное обеспечение, работающее более чем на одной аппаратной платформе и/или операционной системе.

Кроссплатформенные приложения – создаются на языке разметки HTML и стилей CSS, а так же JavaScript. Данные приложения пишутся одновременно для всех платформ и адаптивны к большинству устройств. Большинство разработчиков при разработке кроссплатформенных приложений пользуются фреймворком PhoneGap [13]. Данный фрейморк позволяет получить доступ к аппаратным и программным возможностям платформы.

PhoneGap — бесплатный open-source, созданный Nitobi Software. Позволяет создать приложения для мобильных устройств используя JavaScript, HTML5 и CSS3, без необходимости знания «родных» языков программирования (например Java, Swift). Готовое приложение компилируется в виде установочных пакетов для каждой мобильной операционной системы.

Преимущества кроссплатформенной разработки:

- Экономия бюджета проекта, так как используется одна технология и один набор графики, что позволяет снизить количество рабочих часов;
- Время разработки – отсутствие уникальных элементов интерфейса позволяет значительно снизить сроки разработки.

Недостатки кроссплатформенной разработки:

- Не используются уникальные особенности платформ;
- Медленная работа приложения.

1.2 Техническое задание

1.2.1 Назначение и цели создания мобильного приложения

1.2.1.1 Назначение мобильного приложения

Мобильное приложение – программное средство, предназначенное для работы с сервисом на удобной для пользователя мобильной платформе и выполнения основных действий, возможных в программе в целом.

1.2.1.2 Цель создания мобильного приложения

Основной целью создания приложения является реализация программы лояльности на мобильной платформе. Это является одной из основ данной программы лояльности.

1.2.2 Требования к структуре и функциональной части мобильного приложения

Функциональная структура мобильного приложения должна включать в себя:

- Понятный пользователю интерфейс;
- Загрузка информации из БД;
- Простые и понятные механизмы получения, траты и прочих действий с бонусами;
- Реферальная система;
- Возможность заказа товаров

1.2.3 Требования к видам обеспечения

1.2.3.1 Требования к лингвистическому обеспечению

1.2.3.1.1 Языки программирования

Для разработки приложения использовались следующие языки программирования:

- TypeScript
- JavaScript
- CSS

1.2.3.1.2 Языки взаимодействия пользователей и системы

Основным языком взаимодействия пользователей и системы является русский язык, так же необходимо организовать поддержку иностранных языков, таких как английский.

1.2.3.2 Требования к программному обеспечению

Программное обеспечение должно быть представлено в виде мобильного приложения для Android 4.2 и более и iOS 8 и выше.

1.3 Система Elleum

Elleum [1] является программой лояльности, работающей по следующей схеме:

- Клиент совершает покупки в магазинах, сотрудничающих с Elleum и тем самым получает бонусы на свой счет;
- Бонусы будут копиться в личном кабинете Elleum. Каждый магазин устанавливает процент бонусов самостоятельно. Бонусы со всех магазинов будут аккумулироваться на личном счете пользователя.
- Клиент оплачивает бонусами до 100% от покупки в любых магазинах, подключенных к Elleum.

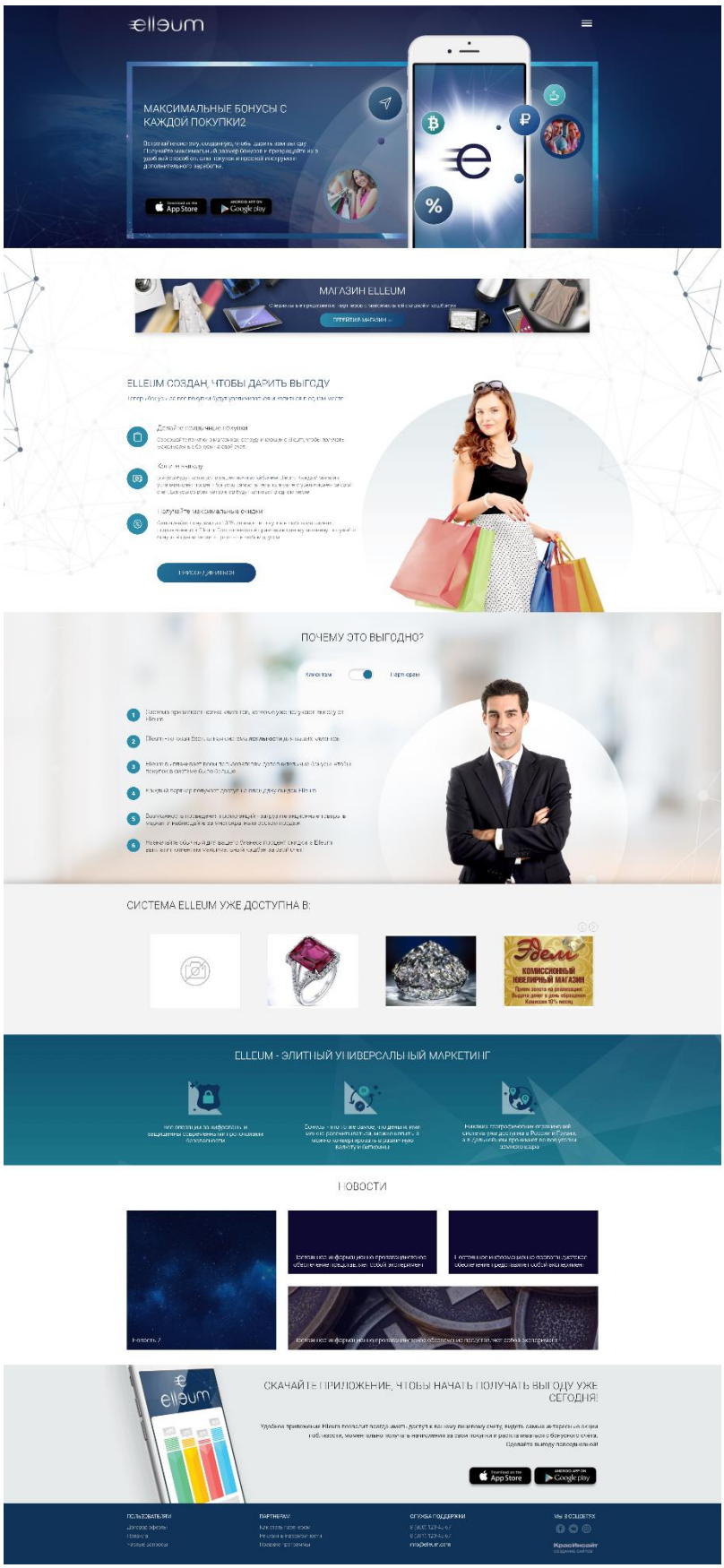

Рисунок 2 – сайт системы Elleum

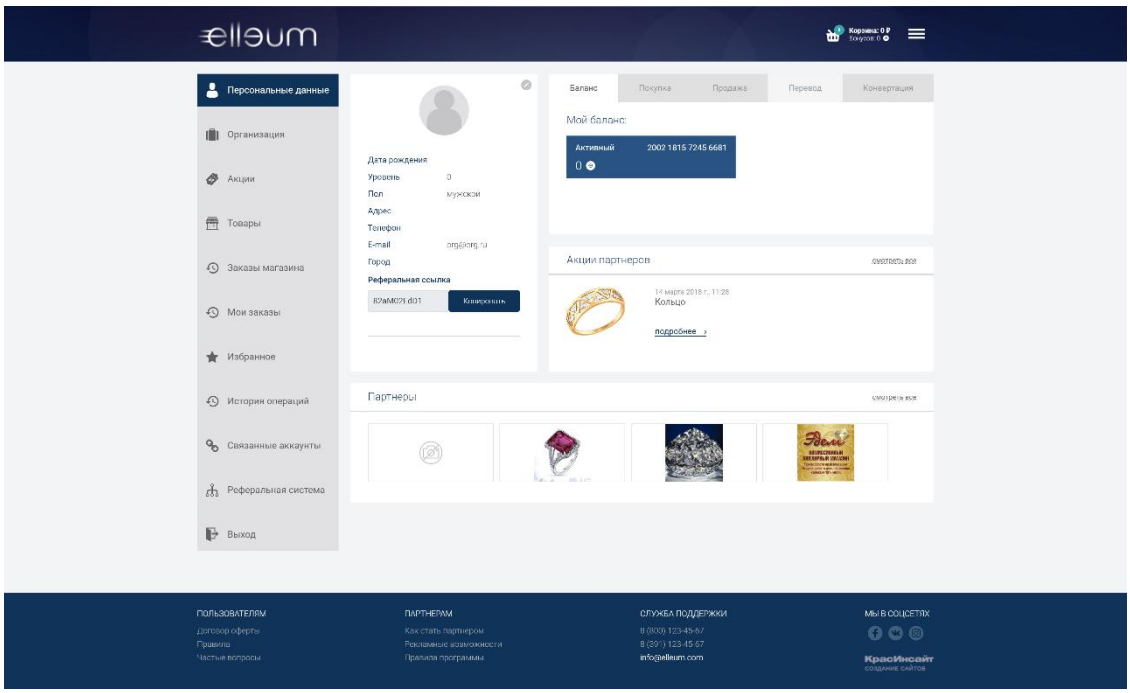

Рисунок 3 – личный кабинет

На сайте системы у каждого пользователя имеется личный кабинет, в котором пользователь может редактировать персональные данные, скопировать свою реферальную ссылку, оперировать с балансом, просматривать историю своих заказов, увидеть некоторые акции, просмотреть свое реферальное дерево. Так же, если пользователь обладает правами организации, то он может регистрировать свои товары и акции. Помимо этого, на сайте имеется аккаунт администратора, который обладает повышенными привилегиями: он может просматривать и влиять на все операции с бонусами, происходящими в системе, создавать и редактировать новости, просматривать всех пользователей системы, администрировать сайт, в частности, изменять такие параметры, как телефон техподдержки, приветственное сообщение и прочее, а так же редактировать раздел часто задаваемых вопросов.

1.4 Анализ существующих систем лояльности

На сегодняшний день существует огромное количество различных видов программ лояльности, поэтому требуется рассмотреть существующие решения, выделить их преимущества и недостатки.

1.4.1 Bon-Bon

Мобильное приложение [2] позволяет создать клиенту собственную QR-карту, по которой он получает бонусы от покупок. При наличии бонусов на счету клиент может использовать их в качестве оплаты за покупку в любом магазине, зарегистрированном в данной системе.

На рисунке 4 представлены окна данного мобильного приложения.

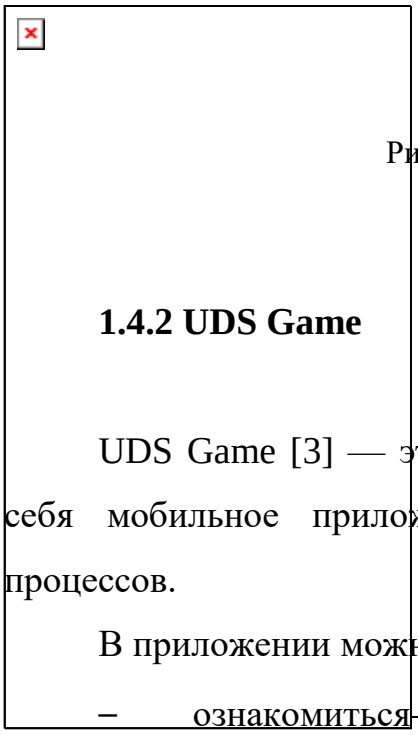

Рисунок 4 – мобильное приложение Bon Bon

1.4.2 UDS Game

UDS Game [3] — это комплексное решение для бизнеса, включающее в себя мобильное приложение и инструменты для оптимизации бизнес-процессов.

В приложении можно:

- ознакомиться с прейскурантом/ меню;
- получить скидку;
- посмотреть местоположение;
- изучить описание и рейтинг заведения.

На рисунке 5 представлены окна данной программы.

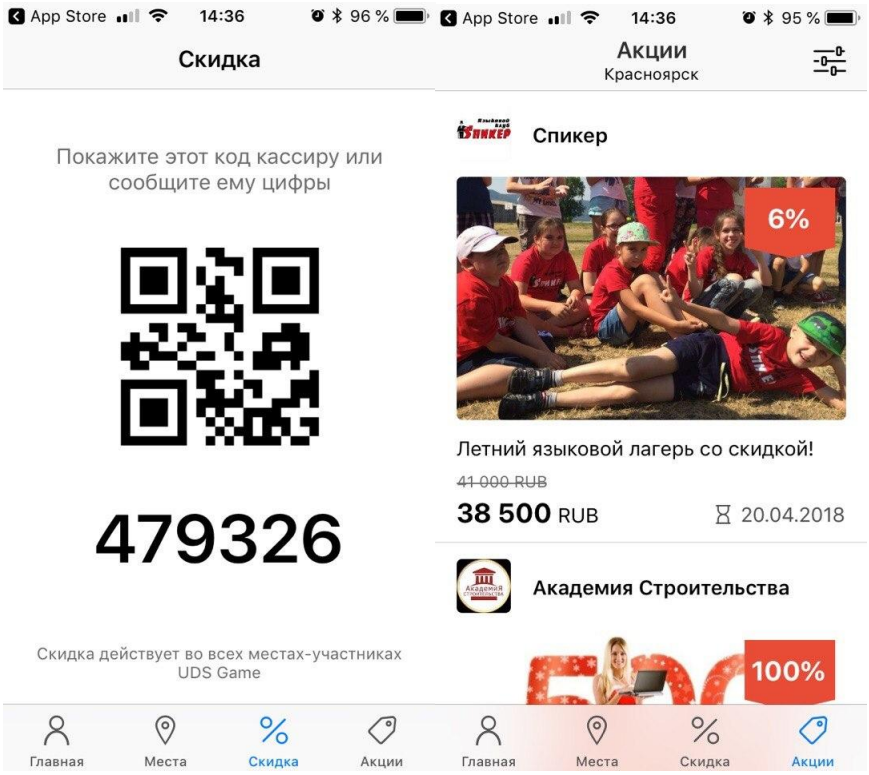

Рисунок 5 – мобильное приложение UDS Game

Вывод: были проанализированы указанные системы и принято решение, что в программе лояльности, помимо реализованных у существующих систем лояльности функций, так же необходимы такие, как:

- ручной перевод средств между пользователями;
- обзор осуществленных заказов;
- доступ к базе ответов на часто задаваемые вопросы.
- реферальная система

1.5 Выбор средств разработки

1.5.1 TypeScriprt

TypeScript [4] — язык программирования, представленный Microsoft в 2012 году и позиционируемый как средство разработки веб-приложений, расширяющее возможности JavaScript.

TypeScript отличается от JavaScript возможностью явного статического назначения типов, поддержкой использования полноценных классов (как в традиционных объектно-ориентированных языках), а также поддержкой подключения модулей, что призвано повысить скорость разработки, облегчить читаемость, рефакторинг и повторное использования кода, помочь осуществлять поиск ошибок на этапе разработки и компиляции, и, возможно, ускорить выполнение программ.

Компилятор TypeScript называется tsc, написан на языке TypeScript и может быть скомпилирован в стандартный JavaScript, и может быть запущен на любом движке JavaScript, например, в браузере. Компилятор идет вместе с сервером сценариев, который может запускать компилятор. Также он доступен в виде пакета для node.js, который использует node.js в качестве сервера.

Язык широко поддерживается множеством IDE, удобен в использовании и является перспективным и может быть с использованием Apache Cordova использоваться как для разработки под Android, так и под iOS.

1.5.2 Apache Cordova

Apache Cordova [5] — это платформа разработки мобильных приложений с открытым исходным кодом. Она позволяет использовать стандартные веб-технологии, такие как HTML5, CSS3 и JavaScript для кросс платформенной разработки, избегая родного языка разработки для каждой из мобильных платформ. Приложения выполняются внутри обертки, нацеленной на каждую

платформу и полагаются на стандартные API для доступа к датчикам устройства, данным и состоянию сети.

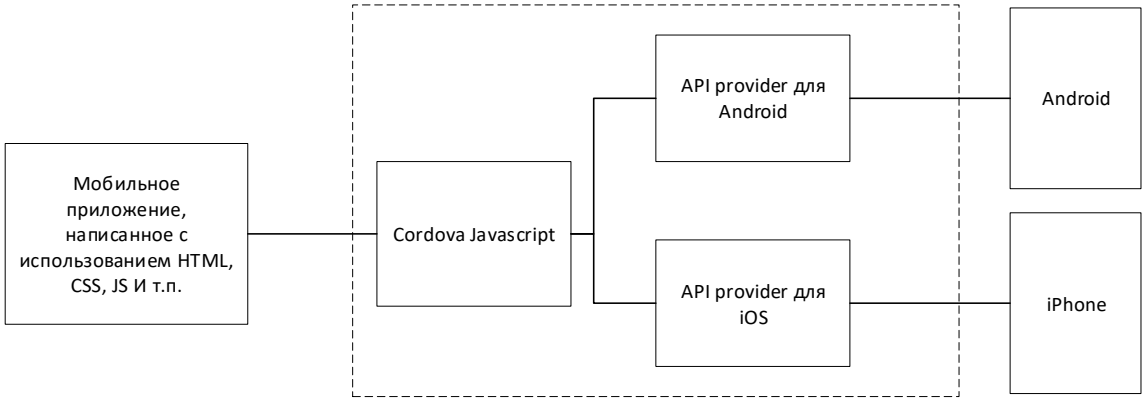

Рисунок 6 – принцип работы Apache Cordova

Именно кроссплатформенность написанных с использованием Apache Cordova приложений является основным фактором в пользу использования при написании клиентской части приложения.

1.5.3 Ionic Framework

Ionic [6] - это полноценный SDK с открытым исходным кодом для разработки мобильных приложений. Оригинальная версия была выпущена в 2013 году и построена на AngularJS и Apache Cordova. Более поздние версии, известные как Ionic 3 или просто «Ionic», построены на Angular. Ionic предоставляет инструменты и услуги для разработки гибридных мобильных приложений с использованием веб-технологий, таких как CSS, HTML5 и Sass.

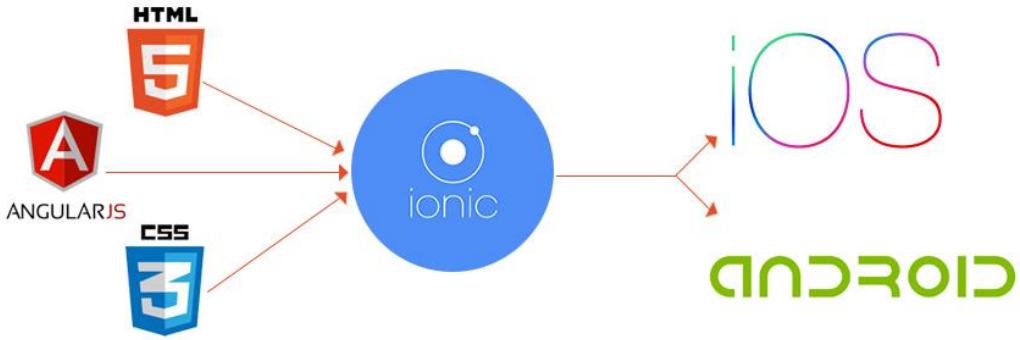

Рисунок 7 – Ionic Framework

Широкий функционал встроенных средств разработки и унаследованная от Cordova кроссплатформенность в купе с бесплатностью Ionic CLI являются основными аргументами в пользу его использования в данной работе.

1.5.4 mySQL

MySQL [7] — свободная реляционная система управления базами данных.

MySQL является решением для малых и средних приложений. Входит в состав серверов WAMP, AppServ, LAMP и в портативные сборки серверов Денвер, XAMPP, VertrigoServ. Обычно MySQL используется в качестве сервера, к которому обращаются локальные или удалённые клиенты, однако в дистрибутив входит библиотека внутреннего сервера, позволяющая включать MySQL в автономные программы. Имеет открытый исходный код, проста в использовании, надежна.

1.5.5 Laravel

Laravel [8] — бесплатный веб-фреймворк с открытым кодом, предназначенный для разработки с использованием архитектурной модели

MVC. Позволяет удобно работать с базами данных, просто и понятно создавать для них API.

1.5.6 Node.js

Node или Node.js [9] — программная платформа, основанная на движке V8 (транслирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API, подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода. Node.js применяется преимущественно на сервере, выполняя роль веб-сервера. В основе Node.js лежит событийно-ориентированное и асинхронное программирование с неблокирующим вводом/выводом.

Серверную часть приложения планируется реализовать именно с использованием Node и его плагинов для работы с базами данных.

1.5.7 JSON

JSON [10] — текстовый формат обмена данными, основанный на JavaScript. Как и многие другие текстовые форматы, JSON легко читается людьми.

Несмотря на происхождение от JavaScript, формат считается независимым от языка и может использоваться практически с любым языком программирования. Для многих языков существует готовый код для создания и обработки данных в формате JSON.

За счёт своей лаконичности формат JSON может использоваться для сериализации сложных структур. Если говорить о веб-приложениях, в таком ключе он уместен в задачах обмена данными как между браузером и сервером.

1.6 Вывод по разделу

В результате анализа задания на дипломное проектирование были сформулированы четкие требования к разрабатываемой системе, было решено, что разрабатываемая система будет получать информацию от сервера для отображения ее в удобном для пользователя виде в мобильном приложении; так же был проведен анализ существующих решений и выделены их недостатки и преимущества. Выделен перечень функций, отсутствующих в аналогичных мобильных приложениях программ лояльности и который необходимо реализовать. Для того, чтобы упростить разработку кроссплатформенного мобильного приложения, в частности избежать нативных языков и писать мобильное приложение сразу под две платформы, был сформулирован стек используемых технологий, в частности TypeScript как основной язык, Apache Cordova, как основное средство, позволяющее писать сразу на 2 платформы, Ionic framework, как инструмент для работы с Cordova и Node.js для реализации серверной части мобильного приложения.

2 Проектирование и программная реализация мобильного приложения

2.1 Проектирование приложения

Выполнение поставленных задач подразумевает разработку мобильного модуля гибридного приложения. Была разработана архитектура и далее выполнена программная реализация.

2.2 Алгоритм взаимодействия сервера с клиентом

Общий алгоритм взаимодействия мобильного приложения и сервера представлен на рисунке 8.

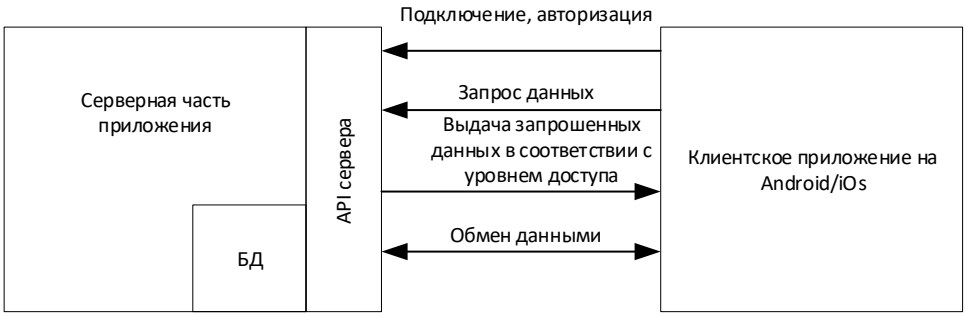

Рисунок 8 –схема взаимодействия мобильного приложения и сервера

Посредством специально разработанного API, клиент подключается к серверу, при этом высылая данные для авторизации. После этого клиент может запросить данные, после чего сервер выдает клиенту сформированный на основе БД и прав клиента JSON. Локально, на мобильном устройстве клиент может что-то сделать с данными, в частности, изменить их. Измененные данные клиент отправляет серверу, где принимается решение, что с ними

делать: занести в БД или игнорировать. Сервер в это время так же может отправлять клиенту новые данные, если такие имеются.

2.3 Формат запроса данных

Механизм формирования запроса является типовым для данного типа приложений и представлен на рисунке 9.

Рисунок 9 – формат запроса

Мобильное приложение-клиент отправляет на сервер url-запрос вида «http://api.elleum.org/api/catalog/categories», на что сервер отвечает клиенту JSON-файлом.

2.4 Структура мобильного приложения

Приложение состоит из множества страниц, написанных на TypeScript, HTML с использованием CSS. Согласно этому, любая страница должна состоять из:

- файла-класса <pagename>.ts, в котором описываются классы и методы, используемы на странице приложением;
- файла-класса <pagename>.module.ts, в котором указаны зависимости этой страницы как модуля программы;
- <pagename>.scss, в котором описаны особые стилевые изменения для страницы;
- <pagename>.html, в котором описана разметка страницы на экране смартфона.

Пример структуры страницы приложения «Заказы» представлен на рисунке 10, код для каждого файла этой страницы – в приложениях А, Б, В, Г соответственно.

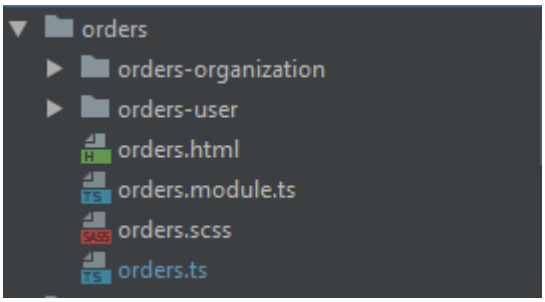

Рисунок 10 – страница «Заказы»

2.5 Разработка архитектуры мобильного приложения

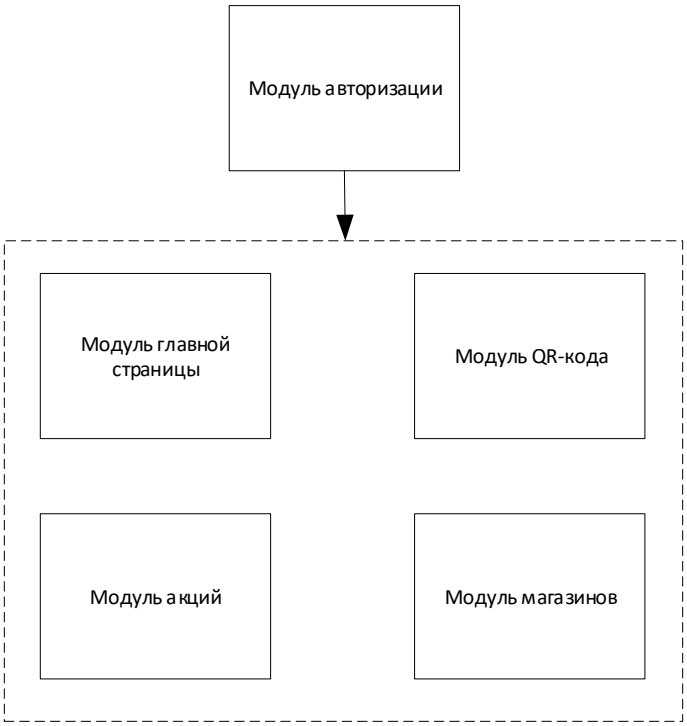

Рисунок 11 – структурная схема мобильного приложения

При разработке алгоритма работы приложения была спроектирована схема, представленная на рисунке 11, из которой можно выделить модули, представленные в следующих подразделах пояснительной записки.

2.6 Модуль авторизации

В данном модуле происходит авторизация пользователя в системе. Так же ему предоставляется возможность зарегистрироваться. После ввода данных в поля «Email» и «Password» приложение отправляет на сервер запрос на авторизацию, на что сервер отвечает либо согласием, либо ошибкой. Если от сервера пришел положительный ответ, приложение переходит на главный экран, на котором отображается основной интерфейс, кнопка профиля, состояние счета и история транзакций.

Следует сказать, что пользователи, в зависимости от их роли в системе, обладают разными правами, в следствие чего у них отличается функционал приложения. Обычный пользователь может делать ряд действий, схожий с действиями обычного пользователя на сайте системы Elleum. Пользователь-организация так же имеет инструментарий для регистрации товаров, работы с акциями. Так же в системе имеется пользователь-администратор, обладающий повышенными привилегиями.

2.7 Основная часть программы

Эта часть приложения является основной. Она состоит из нескольких модулей, между каждым из которых можно переключаться без каких-либо ограничений. У каждого модуля свой функционал, который отличается в зависимости от прав авторизованного пользователя.

2.7.1 Модуль главного экрана

На данном экране выводится основная информация о пользователе; он является отправной точкой для всего приложения после авторизации. Так же тут можно оперировать с балансом, а именно покупать бонусы, переводить их обратно в деньги, переводить бонусы на другой аккаунт. Информация для этого экрана обновляется по запросу к серверу каждый раз, когда экран открывается приложением. Логическая схема данного модуля представлена на рисунке 12.

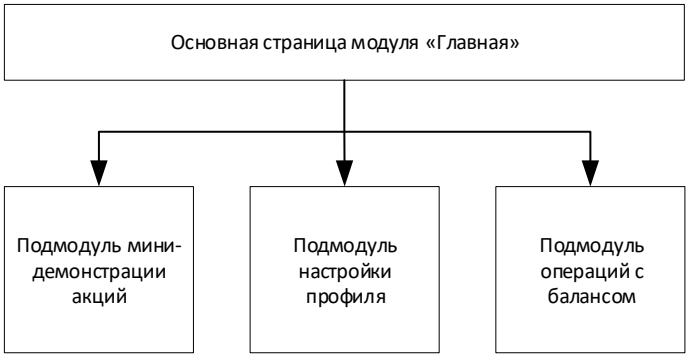

Рисунок 12 – логическая схема модуля «Главная»

2.7.2 Модуль магазинов

На данном экране выводятся участвующие в акции магазины по категориям. Информация для этого экрана обновляется по запросу к серверу каждый раз, когда экран открывается приложением. Так же тут реализован модуль поиска магазина по названию, работающий с БД сервера. Схема данного модуля продемонстрирована на рисунке 13.

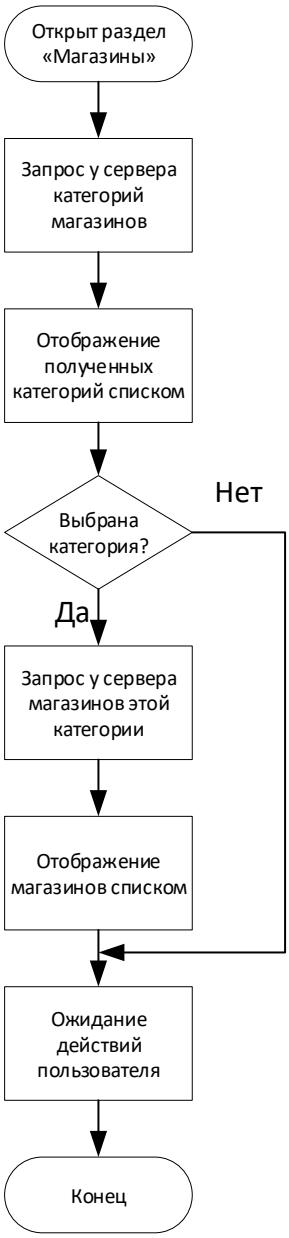

Рисунок 13 – схема модуля «Магазины»

2.7.3 Модуль акций

На данном экране выводятся актуальные акции. Информация для этого экрана обновляется по запросу к серверу каждый раз, когда экран открывается приложением. Схема данного модуля продемонстрирована на рисунке 14.

Рисунок 14 – схема модуля «Акции»

2.7.4 Модуль QR-кода

На данном экране выводится QR-код пользователя. В данной системе QR-код является персональным идентификатором и средством оффлайн-взаимодействия между клиентом и организацией. Генерация QR-кода происходи с использованием qr-image [11] и jsbarcode [12].

2.7.5 Модуль прочего функционала

Тут находится список прочих возможностей, которые реализованы в программе. Для пользователя-организации и пользователя-клиента они различаются. Для клиента тут реализованы магазин, корзина, список заказов, выход из аккаунта. Для организации, здесь же, помимо доступного клиенту – функционал работы с акциями, ручной продажи при помощи QR-кода, товарами и собственным магазином. Логическая схема данного модуля представлена на рисунке 15.

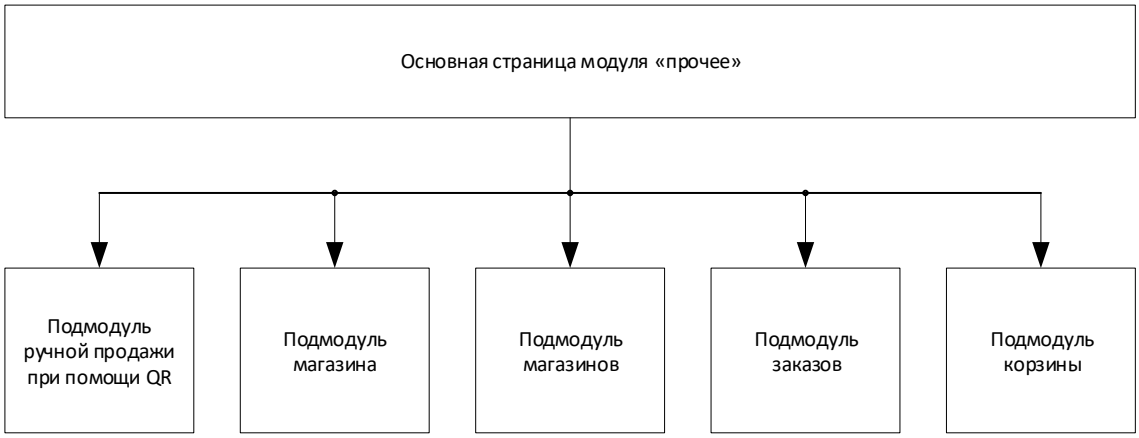

Рисунок .15 – логическая схема модуля «Прочее»

2.8 Bitbucket

Система Elleum разрабатывается командой, каждый член которой работает над отдельной функцией системы: серверной частью, сайтом и мобильным приложением. Для организации внутренней коммуникации и контроля версий был использован Bitbucket, репозиторий проекта на котором показан на рисунке 12

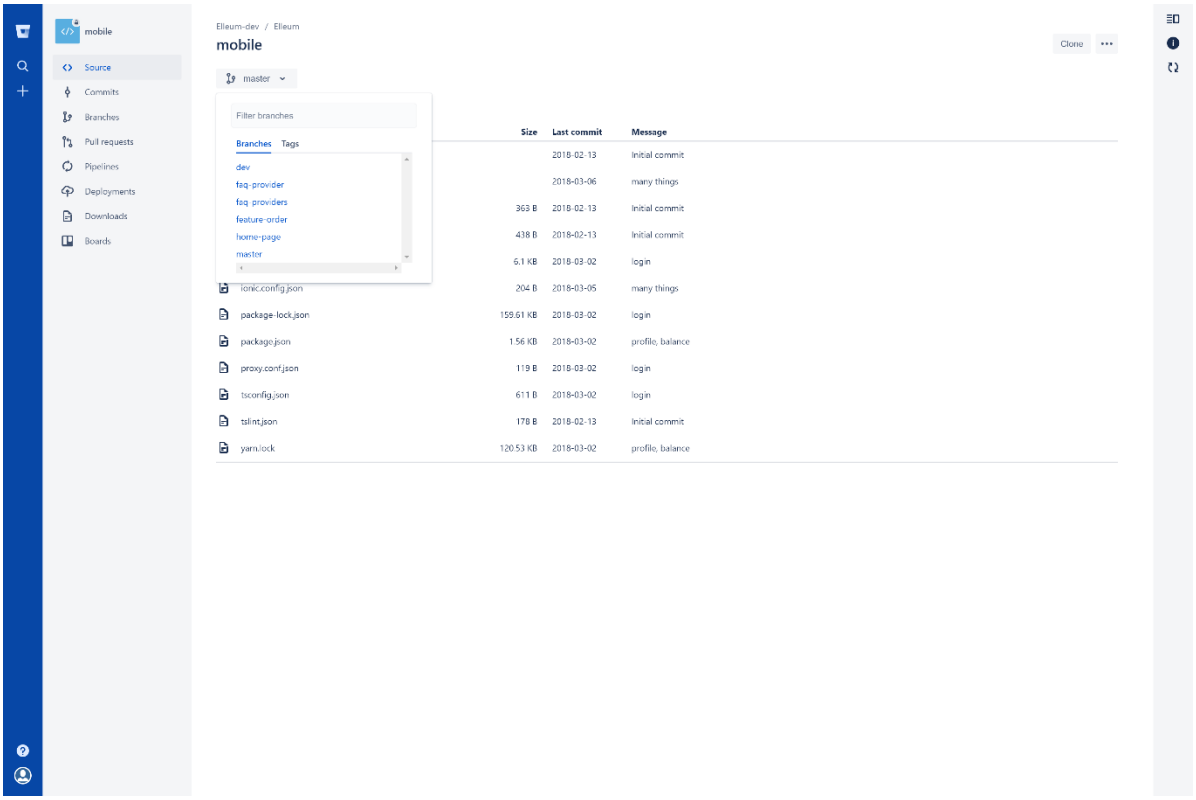

Рисунок 16 – проект на Bitbucket

2.9 Формирование результата разработки

Итог работы –готовое мобильное приложение для платформы Android и iOS. Формирование .apk файла предусмотрено средствами Ionic. Для формирования .ipa файла был использован ресурс Adobe PhoneGap [13], который из сформированных при разработке веб-страниц алгоритмом, схожим с таковым для Android, формирует итоговый .ipa файл. Данное средство было выбрано, поскольку для формирования .ipa файлов необходим компьютер, управляемый Mac OS либо подобный ресурс, что ведет к тому же результату при наименьшем количестве затрат.

Процесс сборки проекта происходит путем загрузки .zip-архива на сайт и дальнейшего формирования .apk и .ipa файла из предоставленного нами кода. Данный процесс показан на рисунке 13.

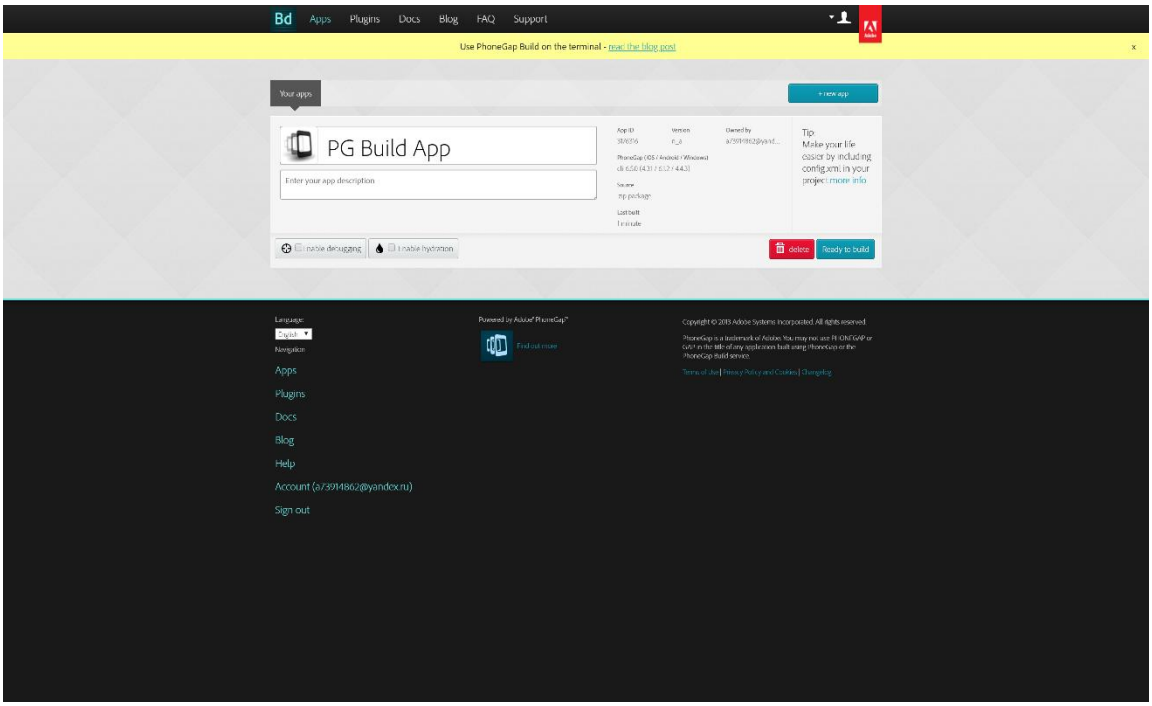

Рисунок 17 – сборка проекта посредством Adobe PhoneGap

3 Описание разработанного приложения

3.1 Установка и запуск

Приложения распространяется посредством Google Play для Android и App Store для iOS. Для установки необходимо зайти на страницу приложения в магазине для своей платформы и скачать, после чего можно запустить приложение, нажав для этого соответствующую иконку на своем смартфоне.

3.2 Работа программы

3.2.1 Раздел «Авторизация»

При включении приложения пользователь увидит окно авторизации, представленное на рисунке 12. Если пользователь зарегистрирован в программе, то ему необходимо заполнить поля «Email» и «Password» актуальными данными, после чего он попадет в основную часть приложения, в частности – на главный экран.

Если же пользователь не зарегистрирован, то необходимо зарегистрироваться, нажав специальную кнопку и заполнив там все поля, после чего выполнить алгоритм для зарегистрированного пользователя.

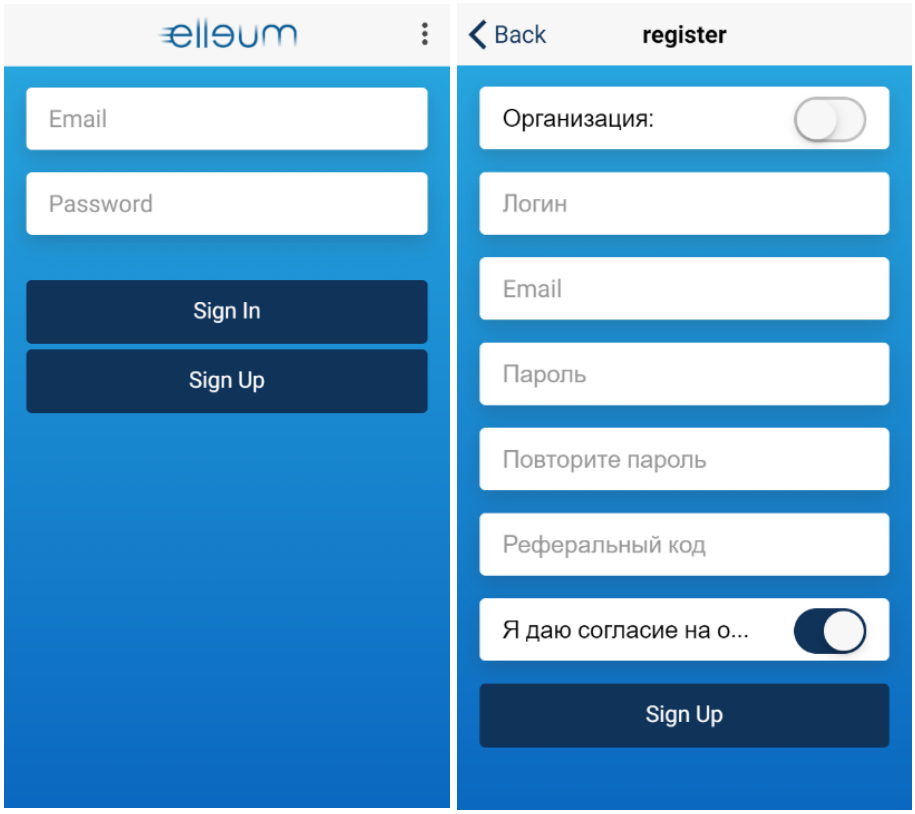

Рисунок 18 – раздел «Авторизация, регистрация»

3.2.2 Раздел «Главная»

На данном экране, представленном на рисунке 13, пользователь может настраивать свой профиль, работать с внутренней валютой, наблюдать историю транзакций, просматривать некоторые активные акции. Для просмотра своего профиля пользователь нажимает на кнопку «Ваш профиль», после чего в открывшемся окне может просмотреть и настроить необходимую информацию.

Для работы с внутренней валютой пользователь нажимает кнопку «…» около номера своего ЛС, после чего в изменившемся поле нажимает на нужную ему иконку.

Так же, помимо уникальных функций данного экрана, пользователь видит общую для всех экранов полоску меню внизу экрана и полоску с логотипом системы и кнопкой доступа к часто задаваемым вопросам.

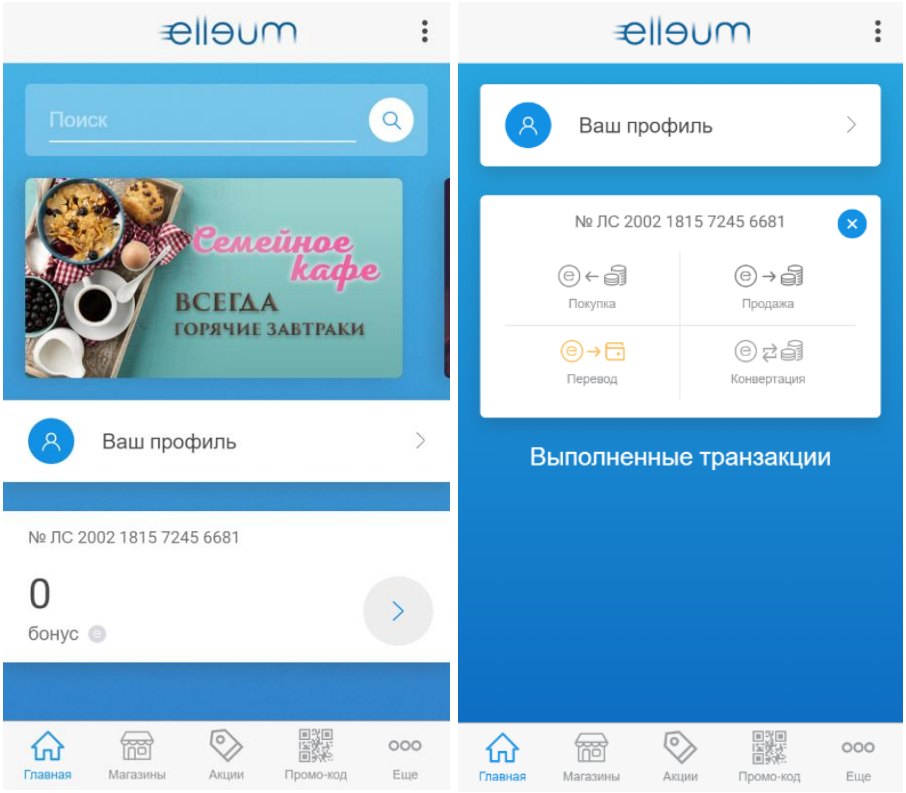

Рисунок 19 – раздел «Главная»

3.2.3 Раздел «Магазины»

На данном экране, представленном на рисунке 14, пользователь может увидеть категории магазинов, участвующих в программе. При нажатии на категорию пользователь увидит отдельные магазины, соответствующие этой категории. Так же, при нажатии на магазин, пользователю будет предоставлена подробная информация о магазине.

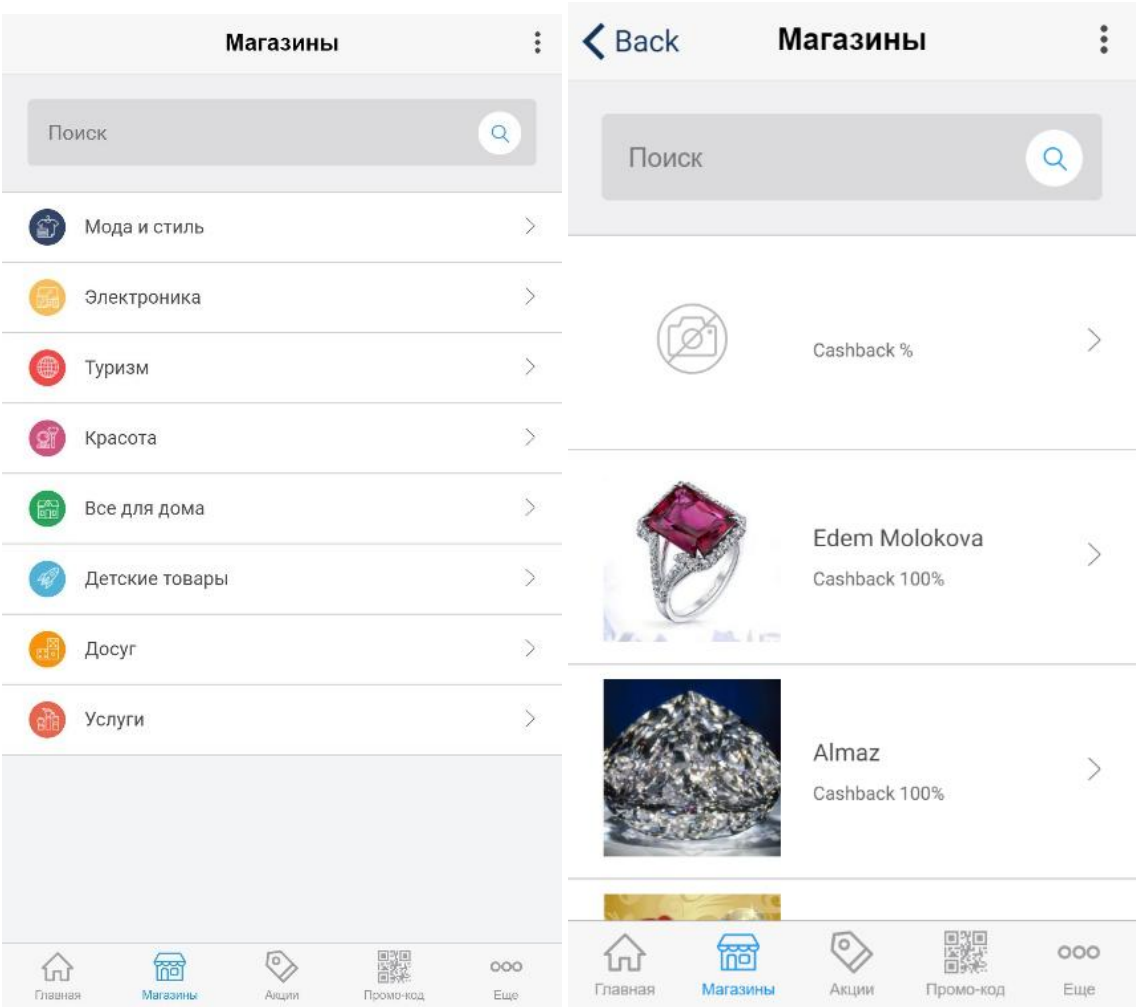

Рисунок 20 – раздел «Магазины»

3.2.4 Раздел «Акции»

На данном экране, представленном на рисунке 15, пользователь может увидеть активные акции, размещенные в системе, а так же, при нажатии на кнопку «Детали», узнать подробности этой акции

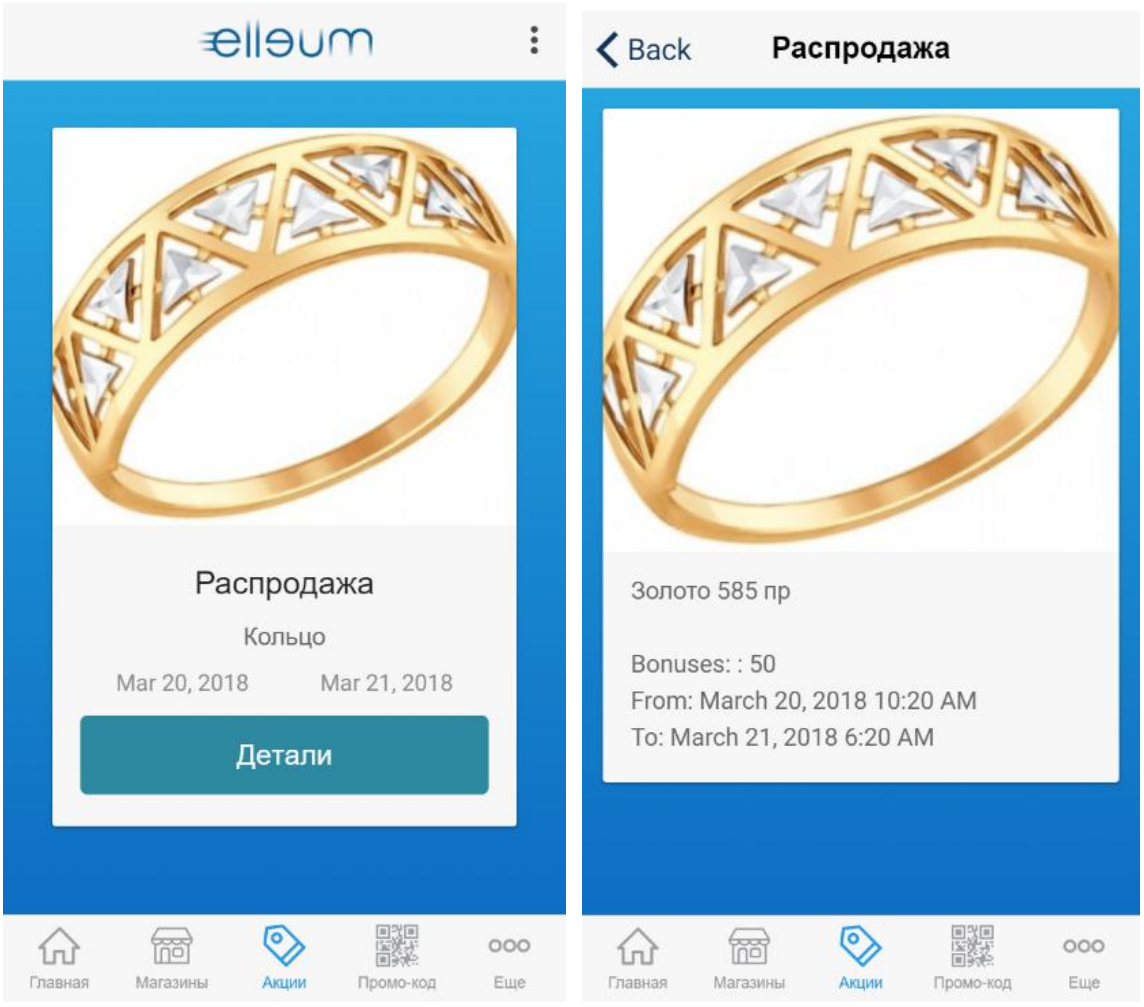

Рисунок 21 – раздел «Акции»

3.2.5 Раздел «Промо-код»

На данном экране, представленном на рисунке 16 пользователь видит свой QR-код, а так же может сгенерировать новый нажатием кнопки «Новый код». Этот модуль позволяет проводить работу в системе даже в режиме оффлайн, путем непосредственного считывания продавцом услуги кода клиента.

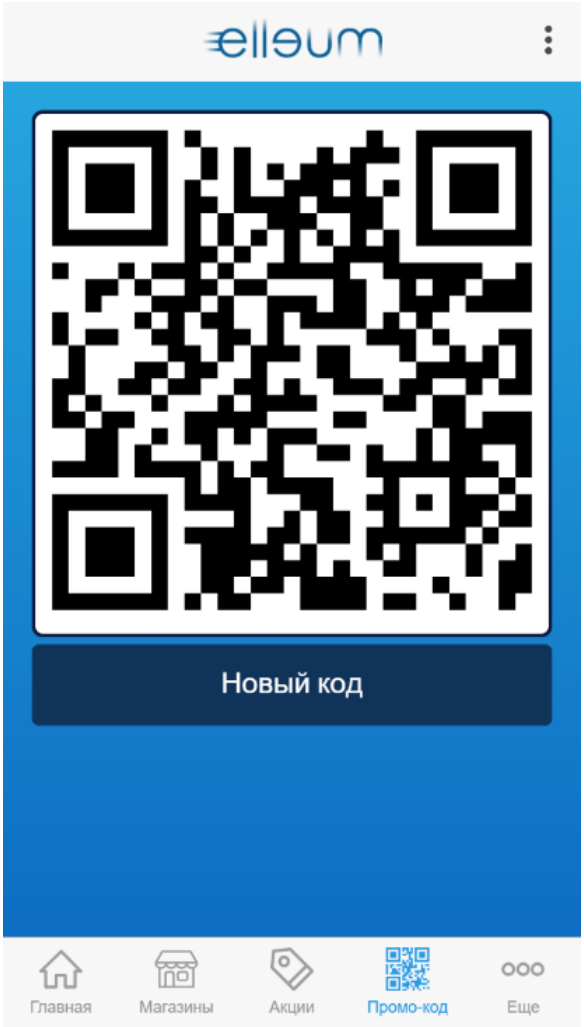

Рисунок 22 – раздел «Промо-код»

3.2.6 Раздел «Еще»

На данном экране, вид которого для организации представлен на рисунке 17, пользователь получает доступ к остальному функционалу программы. У всех пользователей есть кнопка «Магазин», при нажатии на которую открывается внутренний магазин, где предоставлены товары. Покупка товаров происходит нажатием кнопка «Купить» и последующим подтверждением этого в корзине. При нажатии на кнопку «Корзина» пользователь видит те товары, которые он набрал в магазине, но еще пока не заказал. При нажатии на кнопку «Заказы» пользователь видит статус своих заказов, так же может узнать детали о них. При нажатии на кнопку «Выйти» происходит выход из системы и приложение вновь открывает экран авторизации.

Помимо описанного общего функционала, пользователь-организация имеет кнопку «Распродажа», где он может создать акцию, заполнив необходимые поля.

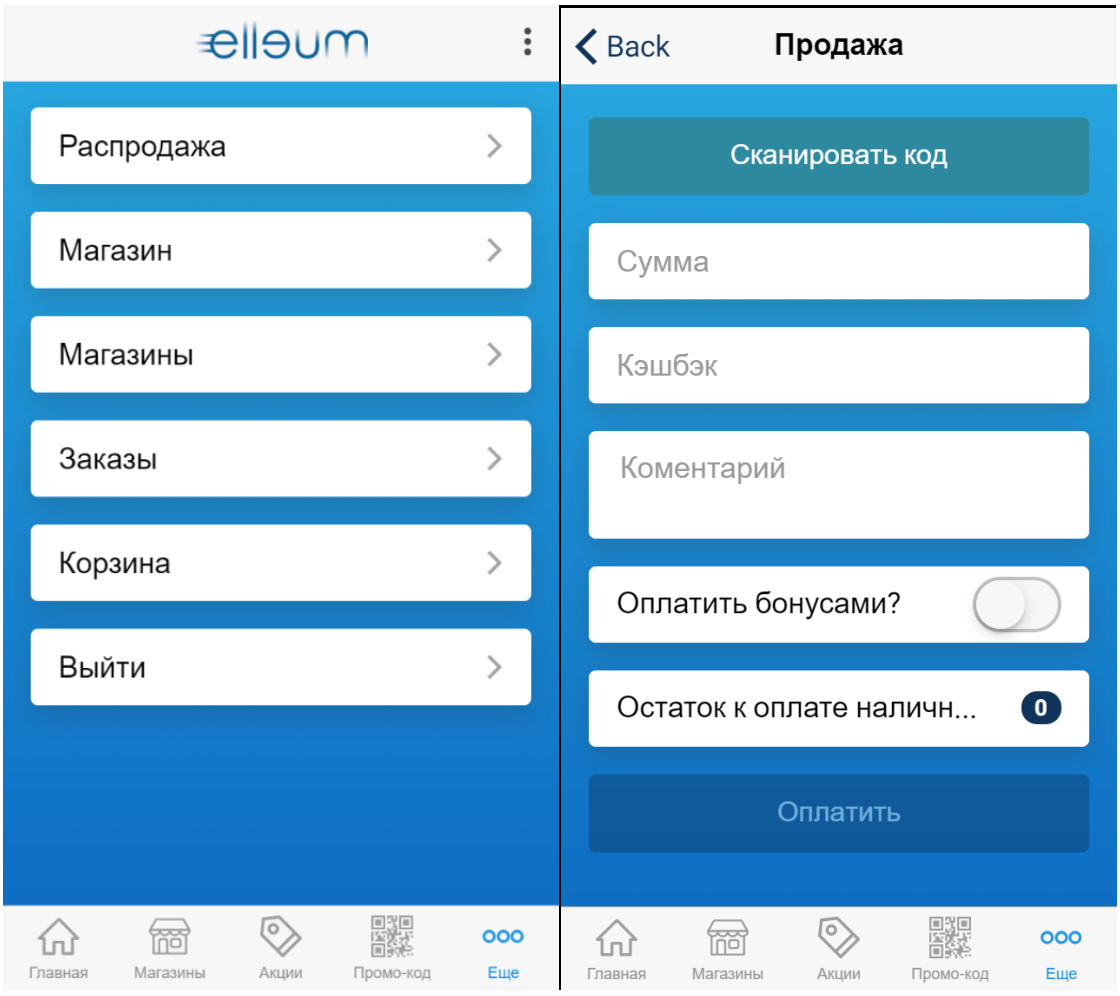

Рисунок 23 – раздел «еще, распродажа»

ЗАКЛЮЧЕНИЕ

В результате выполнения выпускной квалификационной работы была изучена предметная область и существующие на данный момент аналоги. После изучения аналогов был сформулирован ряд требований, предъявляемый к написанному мобильному приложению для программы лояльности.

На основе сформированных требований разработано и протестировано мобильно приложение, которое позволяет работать с системой программы лояльности Elleum как с Android-, так и с iOS-смартфона.

Данное приложение позволяет просматривать магазины, акции, товары, выставлять товары, оперировать с бонусами и деньгами в рамках системы.

Итог всей разработки - .apk и .ipa файлы, предназначенные для публикации мобильного приложения в Google Play и AppStore.

СПИСОК СОКРАЩЕНИЙ

БД – База данных

ЛС – Личный счет

JSON – JavaScript Object Notation

CSS – Cascading Style Sheets

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Elleum [Электронный ресурс] : – Режим доступа: http://elleum.org/
2. Bon Bon – бесплатная бонусная система лояльности – Bon Bon – приложение для телефона [Электронный ресурс] : – Режим доступа: http://bon-bon.info/
3. UDS Game – готовая система лояльности для вашего бизнеса [Электронный ресурс] : – Режим доступа: https://udsgame.com/static/ru/.
4. TypeScript – JavaScript that scales [Электронный ресурс] : – Режим доступа: https://www.typescriptlang.org/
5. Apache Cordova [Электронный ресурс] : – Режим доступа: https://cordova.apache.org/
6. Build Amazing Native Apps and Progressive Web Apps with Ionic Framework and Angular [Электронный ресурс] : – Режим доступа: https://ionicframework.com/
7. MySQL [Электронный ресурс] : – Режим доступа: https://www.mysql.com/
8. Статьи | Laravel по-русски [Электронный ресурс] : – Режим доступа: https://laravel.ru/
9. Node.js [Электронный ресурс] : – Режим доступа: https://nodejs.org/en/
10. JSON [Электронный ресурс] : – Режим доступа: https://www.json.org/json-ru.html
11. qr-image – npm [Электронный ресурс] : – Режим доступа:. https://www.npmjs.com/package/qr-image
12. JsBarcode - Barcode generator written in JavaScrip [Электронный ресурс] : – Режим доступа: http://lindell.me/JsBarcode/
13. PhoneGap [Электронный ресурс] : – Режим доступа:. https://phonegap.com

ПРИЛОЖЕНИЕ А

Исходный код файла orders.ts

```
import {Component, OnInit} from '@angular/core';
import {InfiniteScroll, IonicPage, NavController, NavParams} from 'ionic-angular';
import {Store} from "@ngrx/store";
import {AppState} from "../../core/store";
import {OrderGetOrganizationAll, OrderGetUserAll} from "../../core/store/order/order.actions";
import {Observable} from "rxjs/Observable";
import {IOrder, selectOrderOrganizationAll, selectOrderUserAll} from "../../core/store/order/order.models";
import {Subject} from "rxjs/Subject";

@IonicPage()
@Component({
  selector: 'page-orders',
  templateUrl: 'orders.html',
})
export class OrdersPage implements OnInit {

  public orders$: Observable<any[]>;
  public store$: Observable<any>;
  public ordersUser$: Observable<IOrder[]>;
  public ordersOrganization$: Observable<IOrder[]>;
  public orderOrganizationScrollSubject: Subject<any> = new Subject<any>();
  public orderUserScrollSubject: Subject<any> = new Subject<any>();
  public subpage = 'user';

  public filter1 = {
    page: 1,
    date_from: '',
    date_to: '',
```

```
    status: '',
    limit: '10'
  };

  public filter2 = {
    page: 1,
    date_from: '',
    date_to: '',
    status: '',
    limit: '10'
};

  constructor(public navCtrl: NavController,
              public navParams: NavParams,
              private store: Store<AppState>) {
  }

  ionViewDidLoad() {
    console.log('ionViewDidLoad OrdersPage');
  }

  ngOnInit() {
    this.ordersUser$ = this.store.select(selectOrderUserAll);
    this.ordersOrganization$ = this.store.select(selectOrderOrganizationAll);
    //this.orders$ = this.store.select(selectOrderUserAll);
    this.store$ = this.store.select('order');
    this.fetchUserOrders();
    this.fetchOrganizationOrders();
  }


  fetchUserOrders() {
   this.store.dispatch(new OrderGetUserAll(this.filter1));
  }
  fetchOrganizationOrders() {
```

```
    this.store.dispatch(new OrderGetOrganizationAll(this.filter2));
  }


  doInfiniteUser(infiniteScroll: InfiniteScroll) {
    this.filter1 = {...this.filter1, limit: (+this.filter1.limit + 10).toString()};
    this.fetchUserOrders();
  }

  doInfiniteOrganization(infiniteScroll: InfiniteScroll) {
    this.filter2 = {...this.filter2, limit: (+this.filter2.limit + 10).toString()};
    this.fetchOrganizationOrders();
  }
}
```

ПРИЛОЖЕНИЕ Б

Исходный код файла orders.module.ts

```
import { NgModule } from '@angular/core';
import { IonicPageModule } from 'ionic-angular';
import { OrdersPage } from './orders';
import {PipesModule} from "../../pipes/pipes.module";
import {OrdersOrganizationComponent} from "./orders-organization/orders-organization";
import {OrdersUserComponent} from "./orders-user/orders-user";
import {DirectivesModule} from "../../directives/directives.module";

@NgModule({
  declarations: [
    OrdersPage,
    OrdersOrganizationComponent,
    OrdersUserComponent,
  ],
  imports: [
    IonicPageModule.forChild(OrdersPage),
      PipesModule,
    DirectivesModule,
  ],
})
export class OrdersPageModule {}
```

ПРИЛОЖЕНИЕ В

Исходный код файла orders.scss

```
page-orders {

}
```

ПРИЛОЖЕНИЕ Г

Исходный код файла orders.html

```
<ion-header no-border>

  <ion-navbar>
    <button ion-button menuToggle>
      <ion-icon name="menu"></ion-icon>
    </button>
    <ion-title>orders</ion-title>
  </ion-navbar>

</ion-header>



<ion-content no-padding>
 <div padding>
  <ion-segment *access="'org'" [(ngModel)]="subpage">
   <ion-segment-button value="user">
    User
   </ion-segment-button>
   <ion-segment-button value="organization">
    Organization
   </ion-segment-button>
  </ion-segment>
 </div>
 <div [ngSwitch]="subpage">
  <ng-container *ngSwitchCase="'user'">
   <orders-user [orders]="ordersUser$|async"></orders-user>
  </ng-container>
  <ng-container *ngSwitchCase="'organization'">
   <orders-organization *access="'org'" (scroll)="doInfiniteOrganization($event)"
                        [orders]="ordersOrganization$|async"></orders-organization>
```

```
    </ng-container>
  </div>
</ion-content>
```